The Experience of Heavy Weight Static Analysis of Linux Device Drivers

Alexey Khoroshilov
khoroshilov@linuxtesting.org
Outline

- Heavy Weight Static Analysis
- Linux Driver Verification
- Lessons Learnt
Static Analysis

Key characteristics

• Scope of analysis (kind of bugs)
• False positives (false bugs reported)
• False negatives (real bugs missed)
• Resources required for analysis
Static Analysis: Trade-Off Triangle

- False positives
- Time of analysis
- False negatives
Static Analysis: Trade-Off Triangle

- False positives
- light-weight
- heavy-weight
- Time of analysis
- False negative
Heavy-Weight Analysis

Based on a picture from http://engineer.org.in
Static Analysis vs Model Checking

Static Analysis

Potential bugs found

Model Checking

SAFE
UNSAFE
UNKNOWN

Error trace
Model Checking: Initially

program in C → model of the program → property to be checked → Model Checker → VERDICT: SAFE

ERROR TRACE:
init();
x = 0;
open();
write();
do_write();
if (X == 0)
assert()
Model Checking: Inside

- Reachability problem
Model Checking: Now

- BMC – Bounded Model Checking
- CEGAR – Counter-Example Guided Abstraction Refinement
Bounded Model Checking

- finite unfolding of transition relation
Counter-Example Guided Abstraction Refinement

1. Abstraction
2. Model checking
3. Error trace analysis
4. Model refinement

new precision

The path is unfeasible
There is a path to error state

The path is feasible

program in C

model of the program

SAFE
UNSAFE
We mourn the passing of competition participant Daniel Wonisch. (Feb. 21, 2012)

1st Intl. Competition on Software Verification held at TACAS 2012 in Tallinn, Estonia.

**Motivation**

Competition is a driving force for the invention of new methods, technologies, and tools. This web page describes the competition of software-verification tools, which will take place at TACAS’12.

There are several new and powerful software-verification tools around, but they are very difficult to compare. The reason is that no widely distributed benchmark suite is available and most concepts are only validated in research prototypes. This competition wants to change this.

Only few projects aim at producing stable tools that can be used by people outside the respective development groups, and the development of such tools is not continuous. Also, PhD students and PostDocs do not adequately benefit from tool development because theoretical papers count more...
# SVCOMP'12 Results

<table>
<thead>
<tr>
<th>Competition candidate</th>
<th>BLAST 2.7</th>
<th>CPAchecker ABE 1.0.10</th>
<th>CPAchecker Memo 1.0.10</th>
<th>ESBMC 1.17</th>
<th>FShell 1.3</th>
<th>LLBMC 0.9</th>
<th>Predator 2011/10/11</th>
<th>QARMC-HSF</th>
<th>SATabs 3.0</th>
<th>Wolverine 0.5c</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Affiliation</strong></td>
<td>Moscow, Russia</td>
<td>Passau, Germany</td>
<td>Paderborn, Germany</td>
<td>Southampton, UK</td>
<td>Vienna, Austria</td>
<td>Karlsruhe, Germany</td>
<td>Brno, Czechia</td>
<td>Munich, Germany</td>
<td>Oxford, UK</td>
<td>Princeton, USA</td>
</tr>
<tr>
<td><strong>ControlFlowInteger</strong></td>
<td>71</td>
<td>141</td>
<td>140</td>
<td>102</td>
<td>28</td>
<td>100</td>
<td>17</td>
<td>140</td>
<td>75</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>9900 s</td>
<td>1000 s</td>
<td>3200 s</td>
<td>4500 s</td>
<td>580 s</td>
<td>2400 s</td>
<td>1100 s</td>
<td>4800 s</td>
<td>5400 s</td>
<td>580 s</td>
</tr>
<tr>
<td><strong>DeviceDrivers</strong></td>
<td>72</td>
<td>51</td>
<td>51</td>
<td>63</td>
<td>20</td>
<td>80</td>
<td>80</td>
<td>--</td>
<td>71</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>30 s</td>
<td>97 s</td>
<td>93 s</td>
<td>160 s</td>
<td>3.5 s</td>
<td>1.6 s</td>
<td>1.9 s</td>
<td>--</td>
<td>140 s</td>
<td>65</td>
</tr>
<tr>
<td><strong>DeviceDrivers64</strong></td>
<td>55</td>
<td>26</td>
<td>49</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>1400 s</td>
<td>1900 s</td>
<td>500 s</td>
<td>870 s</td>
<td>0 s</td>
<td>110 s</td>
<td>0 s</td>
<td>--</td>
<td>3200 s</td>
<td>1300 s</td>
</tr>
<tr>
<td><strong>HeapManipulation</strong></td>
<td>--</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>17</td>
<td>20</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 s</td>
<td>16 s</td>
<td>220 s</td>
<td>210 s</td>
<td>1.0 s</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td><strong>SystemC</strong></td>
<td>33</td>
<td>45</td>
<td>36</td>
<td>67</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>57</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4000 s</td>
<td>1100 s</td>
<td>450 s</td>
<td>760 s</td>
<td>2.4 s</td>
<td>630 s</td>
<td>820 s</td>
<td>5000 s</td>
<td>1900 s</td>
<td></td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 s</td>
<td>0 s</td>
<td>270 s</td>
<td>0 s</td>
<td>0 s</td>
<td>--</td>
<td>--</td>
<td>1.4 s</td>
<td>--</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>231</td>
<td>267</td>
<td>280</td>
<td>249</td>
<td>48</td>
<td>206</td>
<td>138</td>
<td>148</td>
<td>236</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>15000 s</td>
<td>4100 s</td>
<td>4300 s</td>
<td>6800 s</td>
<td>580 s</td>
<td>2700 s</td>
<td>1700 s</td>
<td>5600 s</td>
<td>14000 s</td>
<td>3800 s</td>
</tr>
</tbody>
</table>
Outline

- Heavy Weight Static Analysis
- Linux Driver Verification
- Lessons Learnt
Model Checking and Linux Kernel

- Reachability problem
int main(int argc, char* argv[])
{
    ...
    other_func(var);
    void other_func(int v)
    {
        ...
        assert(x != NULL);
    }
    ...
}
Device Driver World

static struct pci_driver DAC960_pci_driver = {
    .name = "DAC960",
    .id_table = DAC960_id_table,
    .probe = DAC960_Probe,
    .remove = DAC960_Remove,
};

static int DAC960_init_module(void)
{
    int ret;

    ret = pci_register_driver(&DAC960_pci_driver);

    if (ret)
    {
        DAC960_init_module_init();
        return ret;
    }

    return ret;
}

module_init(DAC960_init_module);
module_exit(DAC960_cleanup_module);
Pseudo-main generation

```c
int main(int argc, char* argv[]) {
    init_module();
    for(;;) {
        switch(*) {
            case 0: driver_probe(*,*,*); break;
            case 1: driver_open(*,*); break;
            ...
        }
    }
    exit_module();
}
```
Pseudo-main generation (2)

- Order limitation
  - `open()` after `probe()`, but before `remove()`
- Implicit limitations
  - `read()` only if `open()` succeed
- and it is specific for each class of drivers
Model Checking and Linux Kernel

- Reachability problem
Rule Instrumentor

```c
mutex x;

int f(int y)
{
    lock(x);
    ...
    unlock(x);
    return y;
}

int x_locked = 0;

int f(int y)
{
    assert(x_locked == 0);
    x_locked = 1;
    ...
    assert(x_locked == 1);
    x_locked = 0;
    return y;
}
```
Aspect-Oriented Approach

mutex x;

int f(int y)
{
    lock(x);
    ...
    unlock(x);
    return y;
}

Aspect:
around:
call(int lock(mutex x)
{
    assert(x_locked == 0);
    x_locked = 1;
}
Rule Instrumentor

```c
mutex x;
int f(int y)
{
    lock(x);
    ...
    unlock(x);
    return y;
}
```

```c
int x_locked = 0;
int f(int y)
{
    assert(x_locked == 0);
    x_locked = 1;
    ...
    assert(x_locked == 1);
    x_locked = 0;
    return y;
}
```
Rule Instrumentor: Implementation

- **CIF** – C Instrumentation Framework
  - gcc-based aspect-oriented programming tool for C language
  - available at forge.ispras.ru under GPLv3
Where we are

- Static analysis infrastructure
- Front-ends
  - ldv-manager
  - ldv-git
  - ldv-online
Start Verification

on x86_64 architecture

1. Ensure that drivers satisfy the following requirements:
   - The driver is archived using gzip or bzip2 and has one of the following extensions: .tar.bz2, tar.gz, .tgz
   - Archive should contain:
     o Makefile (written to be compiled with the kernel)
     + obj-m is mandatory
     o Sources needed by Makefile
   - Archive should not contain generated files left from builds

2. Upload driver.

3. Wait for results.
Verification Report

Driver: test-0032-wl12xx-unsafe.tar.bz2  
Timestamp: 2011-01-19 20:51:12  
Verification architecture: x86_64

You can see verification verdict for each rule and linux kernel. Verdict may be:

- **Safe** - there is no mistakes for the given linux kernel and rule.
- **Unsafe** - driver may contain an error. You can see the error trace by clicking on the "Unsafe" link for the corresponding linux kernel and rule.
- **Build failed** - your driver is not compatible with the given linux kernel. In this case you may see the compile error trace by clicking on the "more details" link.
- **Unknown** - tools can not determine whether your driver Safe or Unsafe.
- **Queued** - the driver waits for the turn to verification.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutex lock/unlock</td>
<td>Unsafe</td>
</tr>
<tr>
<td>NIO allocation under usb lock</td>
<td>Safe</td>
</tr>
<tr>
<td>Module get/put</td>
<td></td>
</tr>
<tr>
<td>PCI pool create/destroy, alloc/tree</td>
<td></td>
</tr>
<tr>
<td>Delay in probe io, on/off</td>
<td></td>
</tr>
<tr>
<td>Memory allocation inside spinlocks</td>
<td></td>
</tr>
<tr>
<td>Linked list double add</td>
<td></td>
</tr>
<tr>
<td>Usb alloc/free urb</td>
<td></td>
</tr>
<tr>
<td>Spinlocks lock/unlock</td>
<td></td>
</tr>
</tbody>
</table>
Where we are

- Static analysis infrastructure
- Cluster framework
- Front-ends
  - ldv-manager
  - ldv-git
  - ldv-online
- Results database
  - Error trace visualizer
  - Knowledge base
  - Comparison framework
Error Trace Visualizer

Rule: Mutex lock/unlock

```
3182   LDV_IN_INTERRUPT = 1;
3191   +ldv_initialize_FOREACH();
3195   tmp_8 = nondet_int(); /* The function body
3198   assert(tmp_8 != 0);
3200   tmp_7 = nondet_int(); /* The function body
3208   assert(tmp_7 != 0);
3280   assert(tmp_7 != 1);
3360   assert(tmp_7 != 2);
3440   assert(tmp_7 != 3);
3520   assert(tmp_7 != 4);
3600   assert(tmp_7 != 5);
3680   assert(tmp_7 != 6);
3760   assert(tmp_7 != 7);
3840   assert(tmp_7 != 8);
3920   assert(tmp_7 != 9);
4000   assert(tmp_7 != 10);
4080   assert(tmp_7 != 11);
4130   -carl9170_op_set_key(var_group) /* hw */
        carl9170_op_set_key(var_group) /* hw */
        carl9170_op_set_key(var_group) /* hw */

3182   +ldv_initialize_FOREACH();
3191   tmp_8 = nondet_int(); /* The function body
3198   assert(tmp_8 != 0);
3200   tmp_7 = nondet_int(); /* The function body
3208   assert(tmp_7 != 0);
3280   assert(tmp_7 != 1);
3360   assert(tmp_7 != 2);
3440   assert(tmp_7 != 3);
3520   assert(tmp_7 != 4);
3600   assert(tmp_7 != 5);
3680   assert(tmp_7 != 6);
3760   assert(tmp_7 != 7);
3840   assert(tmp_7 != 8);
3920   assert(tmp_7 != 9);
4000   assert(tmp_7 != 10);
4080   assert(tmp_7 != 11);
```

```
1026   static int carl9170_op_set_key(struct ieee80211_hw *hw, enum set_key_c
1027         struct ieee80211_vif *vif,
1028         struct ieee80211_sta *sta,
1029         struct ieee80211_key_conf *key)
1030   {
1031   struct ar9170 *ar = hw->priv;
1032   int err = 0, 1;
1033   u8 ktype;
1034   if (ar->disable_offload || !vif)
1036   return -EOPNOTSUPP;
1038   /* We have to fall back to software encryption, whenever
1040   the user choose to participates in an IBSS or is connected
1042   to more than one network.
1043   * This is very unfortunate, because some machines cannot handle
1044   * the high throughtput speed in 802.11n networks.
1046   if (!is_main_wif(ar, vif))
1047   goto err_softw;
1050   /* While the hardware supports *catch-all* key, for offloading
1052   group-key en-/de-cryption. The way of how the hardware
1053   decides which keyId maps to which key, remains a mystery...
```
## Knowledge Base

<table>
<thead>
<tr>
<th>#</th>
<th>Task</th>
<th>Kernel</th>
<th>Rule</th>
<th>Total</th>
<th>Safe</th>
<th>Unsafe</th>
<th>Unknown</th>
<th>Verdicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Task description</td>
<td>linux-2.6.38.2</td>
<td>Fail before RI</td>
<td>75</td>
<td>-</td>
<td>-</td>
<td>75</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>May, 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>32_1a</td>
<td>2747</td>
<td>2077</td>
<td>66</td>
<td>604</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>32_7</td>
<td>2747</td>
<td>2227</td>
<td>20</td>
<td>500</td>
<td>3</td>
<td>13</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>39_7</td>
<td>2747</td>
<td>2244</td>
<td>15</td>
<td>488</td>
<td>2</td>
<td>11</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>68_1</td>
<td>2747</td>
<td>2129</td>
<td>68</td>
<td>550</td>
<td>2</td>
<td>26</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Task description</td>
<td>linux-2.6.39</td>
<td>Fail before RI</td>
<td>81</td>
<td>-</td>
<td>-</td>
<td>81</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>June 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>32_7</td>
<td>2826</td>
<td>2278</td>
<td>21</td>
<td>527</td>
<td>2</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>Task description</td>
<td>linux-2.6.39</td>
<td>Fail before RI</td>
<td>81</td>
<td>-</td>
<td>-</td>
<td>81</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>August 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>08_1</td>
<td>2826</td>
<td>2124</td>
<td>50</td>
<td>652</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>32_7</td>
<td>2826</td>
<td>2292</td>
<td>29</td>
<td>505</td>
<td>5</td>
<td>24</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>39_7</td>
<td>2826</td>
<td>2319</td>
<td>19</td>
<td>488</td>
<td>4</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>43_1a</td>
<td>2826</td>
<td>1861</td>
<td>7</td>
<td>958</td>
<td>1</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>68_1</td>
<td>2826</td>
<td>2186</td>
<td>76</td>
<td>564</td>
<td>2</td>
<td>28</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>Task description</td>
<td>linux-3.0.1</td>
<td>Fail before RI</td>
<td>102</td>
<td>-</td>
<td>-</td>
<td>102</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>August 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>08_1</td>
<td>3203</td>
<td>2550</td>
<td>66</td>
<td>587</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>32_7</td>
<td>3203</td>
<td>2631</td>
<td>43</td>
<td>529</td>
<td>11</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>39_7</td>
<td>3203</td>
<td>2659</td>
<td>24</td>
<td>520</td>
<td>5</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>43_1a</td>
<td>3203</td>
<td>2623</td>
<td>8</td>
<td>572</td>
<td>1</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>68_1</td>
<td>3203</td>
<td>2524</td>
<td>90</td>
<td>589</td>
<td>3</td>
<td>58</td>
<td>1</td>
</tr>
</tbody>
</table>
Bugs Found

http://linuxtesting.org/results/ldv

- 50 patches already applied

### Problems in Linux Kernel

This section contains information about problems in Linux kernel found within [Linux Driver Verification](http://linuxtesting.org/results/ldv) program.

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Brief</th>
<th>Added on</th>
<th>Accepted</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0009</td>
<td>Leak</td>
<td>(ath5k) sc-&gt;ah is allocated in ath5k_init_SOFTC() but is not freed</td>
<td>2011-08-08</td>
<td>Kernel Bug Tracker, [bug #37592](<a href="https://lkml.org/lkml/2011/8/23/380">https://lkml.org/lkml/2011/8/23/380</a> commit)</td>
<td>Fixed in the kernel 3.1-rc1</td>
</tr>
<tr>
<td>L0045</td>
<td>Leak</td>
<td>drivers/net/wan/farsync.c: module_get() without module_put()</td>
<td>2011-06-20</td>
<td>[<a href="https://lkml.org/lkml/2011/6/17/320">https://lkml.org/lkml/2011/6/17/320</a> commit d0fd64c](<a href="https://lkml.org/lkml/2011/6/17/320">https://lkml.org/lkml/2011/6/17/320</a> commit d0fd64c)</td>
<td>Fixed in kernel</td>
</tr>
</tbody>
</table>
Outline

- Heavy Weight Static Analysis
- Linux Driver Verification
- Lessons Learnt
Lessons Learnt

- Language features support

No matter which advanced techniques implemented by a tool if it does not work on your code
Lessons Learnt

- Language features support
- Efficiently ignore irrelevant details

Ten of thousands irrelevant transitions vs. dozens of relevant ones
Lessons Learnt

- Language features support
- Efficiently ignore irrelevant details
- No premature UNKNOWN

Error: Unsupported C feature (recursion) in line 60858:

tmp = gma_power_begin( tmp24, tmp25);
(CallstackTransferRelation.getAbstractSuccessors)

Bug Finder vs. Safe Prover
Lessons Learnt

- Language features support
- Efficiently ignore irrelevant details
- No premature UNKNOWN
- Pointer analysis is a weak point

complex data structures
containerof
even arrays
many false positives for complex rules
Lessons Learnt

- Language features support
- Efficiently ignore irrelevant details
- No premature UNKNOWN
- Pointer analysis is a weak point
- Engineering Matters
BLAST is a software model checker for C programs. It uses counterexample-driven automatic abstraction refinement to construct an abstract model which is model checked for safety properties.
ISPRAS BLAST 2.6 Release Notes

Speedup ranges from **8 times** on small-sized programs to **30 times** on medium-sized programs

- Logarithmic algorithm for useful-blocks (significantly speedup of trace analysis)
- Improved integration with SMT solvers
  - efficient string concatenation
  - caching of converted formulae
  - optimization of CVC3 options for BLAST use cases
- Formulae normalization moved to solvers since solvers do it faster
- Alias analysis speedup
  - must-aliases are handled separately and faster than may-aliases
  - removed unnecessary debug prints from alias iteration (even a check for debug flag impacts performance significantly in hot places)
- BLAST-specific tuning of OCaml virtual machine options
## SVCOMP'12 Results

<table>
<thead>
<tr>
<th>Competition</th>
<th>BLAST 2.7</th>
<th>CPAchecker ABE 1.0.10</th>
<th>CPAchecker Memo 1.0.10</th>
<th>ESBMC 1.17</th>
<th>FShell 1.3</th>
<th>LLBMC 0.9</th>
<th>Predator 20111011</th>
<th>QARMC-HSF</th>
<th>SATabs 3.0</th>
<th>Wolverine 0.5c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affiliation</td>
<td>Moscow, Russia</td>
<td>Passau, Germany</td>
<td>Paderborn, Germany</td>
<td>Southampton, UK</td>
<td>Vienna, Austria</td>
<td>Karlsruhe, Germany</td>
<td>Brno, Czechia</td>
<td>Munich, Germany</td>
<td>Oxford, UK</td>
<td>Princeton, USA</td>
</tr>
<tr>
<td>ControlFlowInteger</td>
<td>71 9000 s</td>
<td>141 1000 s</td>
<td>140 3200 s</td>
<td>102 4500 s</td>
<td>28 580 s</td>
<td>100 2400 s</td>
<td>17 1100 s</td>
<td>140 4800 s</td>
<td>75 5400 s</td>
<td>39 580 s</td>
</tr>
<tr>
<td>DeviceDrivers</td>
<td>72 30 s</td>
<td>51 97 s</td>
<td>51 93 s</td>
<td>63 160 s</td>
<td>20 3.5 s</td>
<td>80 1.6 s</td>
<td>80 1.9 s</td>
<td>--</td>
<td>71 140 s</td>
<td>68 65 s</td>
</tr>
<tr>
<td>DeviceDrivers64</td>
<td>55 1400 s</td>
<td>26 1900 s</td>
<td>49 500 s</td>
<td>10 870 s</td>
<td>0 0 s</td>
<td>1 110 s</td>
<td>0 0 s</td>
<td>--</td>
<td>32 3200 s</td>
<td>16 1300 s</td>
</tr>
<tr>
<td>HeapManipulation</td>
<td>-- 4 16 s</td>
<td>4 16 s</td>
<td>1 220 s</td>
<td>--</td>
<td>17 210 s</td>
<td>--</td>
<td>20 1.0 s</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>SystemC</td>
<td>33 4000 s</td>
<td>45 1100 s</td>
<td>36 450 s</td>
<td>67 760 s</td>
<td>--</td>
<td>8 2.4 s</td>
<td>21 630 s</td>
<td>8 820 s</td>
<td>57 5000 s</td>
<td>36 1900 s</td>
</tr>
<tr>
<td>Concurrency</td>
<td>-- 0 0 s</td>
<td>0 0 s</td>
<td>6 270 s</td>
<td>0 0 s</td>
<td>--</td>
<td>0 0 s</td>
<td>--</td>
<td>1 1.4 s</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Overall</td>
<td>231 15000 s</td>
<td>267 4100 s</td>
<td>280 4300 s</td>
<td>249 6800 s</td>
<td>48 580 s</td>
<td>206 2700 s</td>
<td>138 1700 s</td>
<td>148 5600 s</td>
<td>236 14000 s</td>
<td>159 3800 s</td>
</tr>
<tr>
<td>Total Files</td>
<td>277</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>Total Score</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
<td>435</td>
</tr>
</tbody>
</table>

- **BLAST 2.7**: Moscow, Russia
- **CPAchecker ABE 1.0.10**: Passau, Germany
- **CPAchecker Memo 1.0.10**: Paderborn, Germany
- **ESBMC 1.17**: Southampton, UK
- **FShell 1.3**: Vienna, Austria
- **LLBMC 0.9**: Karlsruhe, Germany
- **Predator 20111011**: Brno, Czechia
- **QARMC-HSF**: Munich, Germany
- **SATabs 3.0**: Oxford, UK
- **Wolverine 0.5c**: Princeton, USA

- **ControlFlowInteger**: 93 files, max score: 144
- **DeviceDrivers**: 59 files, max score: 103
- **DeviceDrivers64**: 41 files, max score: 66
- **HeapManipulation**: 14 files, max score: 24
- **SystemC**: 62 files, max score: 87
- **Concurrency**: 8 files, max score: 11
- **Overall**: 277 files, max score: 435
Conclusions

- Language features support
- Efficiently ignore irrelevant details
- No premature UNKNOWN
- Pointer analysis is a weak point
- Engineering Matters
Thank you!

Alexey Khoroshilov
khoroshilov@linuxtesting.org
http://linuxtesting.org/project/ldv